maXecurity™ Product Suite

Technical Writing Example

Deployment Guide

Firmware v2.2

Table of Contents

INTRODUCTION 3

Understanding the Functional Architecture of maXecurity 3

APPROACHES TO INTEGRATING MAXECURITY WITH YOUR EXISTING INFRASTRUCTURE 6

Approach 1: Web Server Access Control Lists 7
Approach 2: Using maXecurity as an Application-layer Firewall 8
Approach 3: Network Firewalling 9
Approach 4: Hybrid Application-layer and Network Firewalling 10
Securing the Policy Store 11

ADVANCED DEPLOYMENT TECHNIQUES 12

Enhancing Security with Multiple Authentication Methods 12
Flexible Authorizations with Multiple Security Zones 13

INTEGRATION WITH 3RD-PARTY WEB APPLICATIONS 14

Integrating with 3rd-party Web Application APIs 14
Integrating with 3rd-party Web Applications without APIs 16
Learning more 17

Technical Writing Example

Introduction

Web-based applications provide instant access to the tools and information necessary for your company to function efficiently. But as more and more sensitive data is made available to employees and customers using the web browser, security technologies like web access management become increasingly important. The maXecurity product suite from P2 Security works with your company's infrastructure to prevent unauthorized access to sensitive data.

This guide discusses exactly how maXecurity ensures the security of your protected resources, as well as four approaches to help you deploy maXecurity appliances within your existing company infrastructure. It also details some of the special features built into the maXecurity product suite, including Single Sign-on, TrustMapping and Trusted Cookies. Taking advantage of these features will allow you to incorporate robust security enhancements into new or existing web applications.

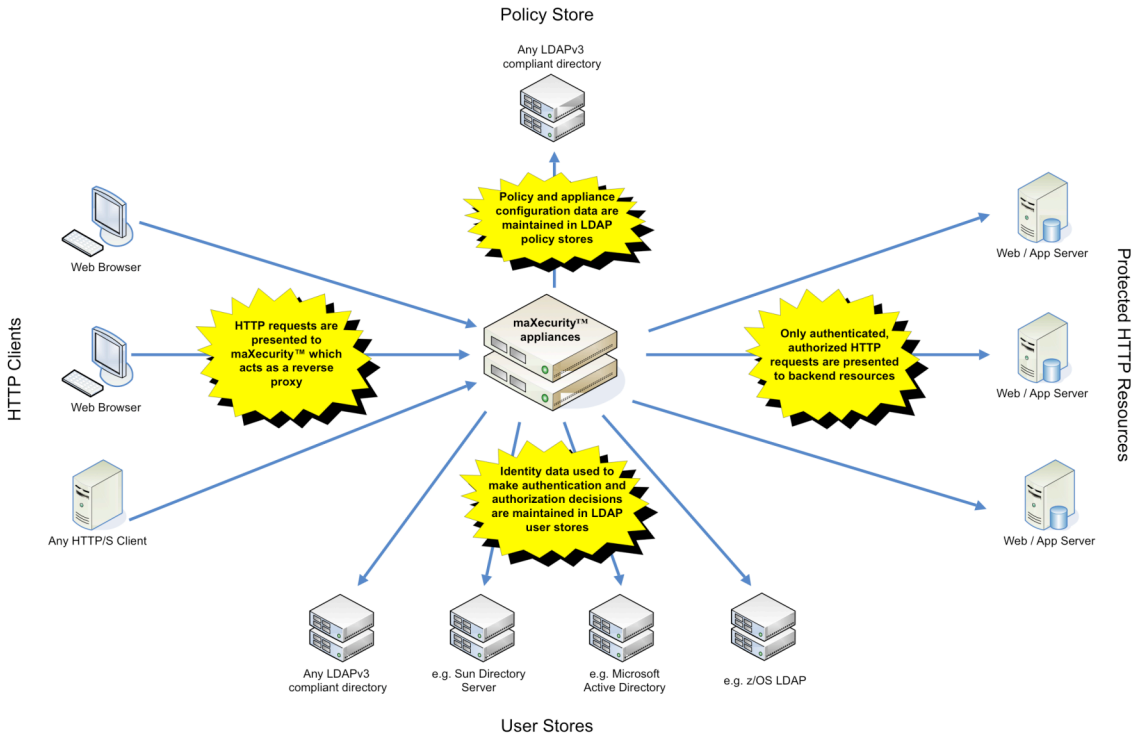
Technical Writing Example

Understanding the Functional Architecture of maXecurity

The maXecurity appliances take a central place in your web-based environment, managing access into and out of your systems. The appliances can also double as a firewall or DMZ for your web and application servers. To scale the system to function within your company infrastructure, you only need to add more maXecurity appliances. Each appliance works in conjunction with your existing infrastructure products, meaning seamless integration with any DNS, router, or switch-based load-balancing or failover product.

Based on your network traffic and the quantity of users you serve, a maXecurity representative can recommend the ideal setup and number of appliances necessary for your environment. As your needs change, you can shuffle or add maXecurity appliances to react to shifting network demands.

This following diagram shows how maXecurity appliances integrate into your existing infrastructure:



Functional architecture diagram. Notice that the maXecurity appliance fits into the center of the network environment. All HTTP requests pass through the appliance before reaching your web/application servers.

Let's examine each part of the functional architecture to see how the maXecurity appliances enforce your company's security policy.

HTTP Clients – On the left of the diagram, HTTP client refers to the web browser—or any HTTP or HTTPS client—that makes a web request. To ensure the security of your web servers and applications, *all* HTTP requests must go through the maXecurity appliance before being passed on to your protected web servers and applications (on the right of the diagram).

Policy Store – The policy store is an LDAPv3-compliant directory that will store information about which domains to protect, credentials to log in, and other configuration information. When the maXecurity appliance boots up, it connects immediately to the policy store using simple credentials (name and password) or digital certificates. Because all appliances refer to the same policy store, you will never need to worry about synchronizing settings from one machine to another.

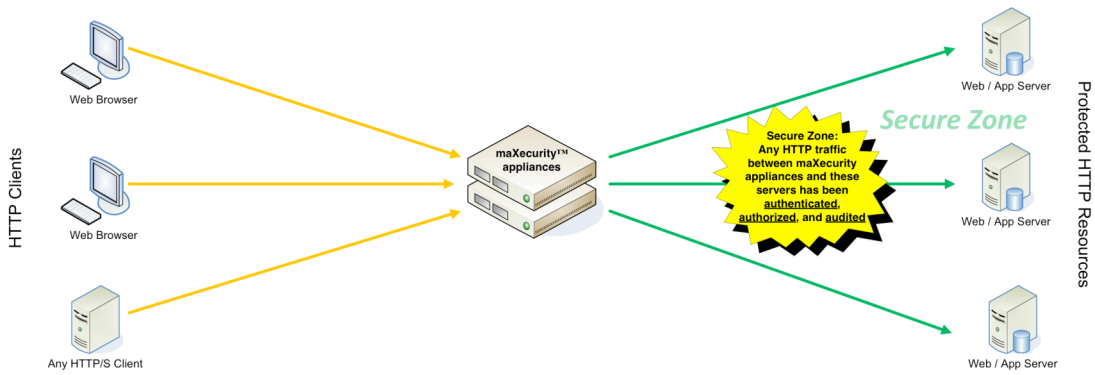
User Stores – A user store is a repository of information about users and groups stored on an LDAP server. All maXecurity products support industry-standard

Technical Writing

LDAP servers, including Microsoft Active Directory, Novell eDirectory, Sun Directory Server, and RACF (with z/OS LDAP). User stores that you configure are available for all domains for authentication and authorization purposes.

HTTP Protected Resources (Domains) – On the far right of the diagram are your company's back-end web servers and applications. Because all HTTP requests must pass through the maXecurity appliance, only authenticated, authorized web requests will reach your protected resources.

In summary, maXecurity appliances stand between the web browser and your company's web servers. They listen for HTTP requests and allow only authorized users to gain access to your company's secure data and applications.



The Secure Zone. All HTTP traffic between the maXecurity appliances and your company's back-end web/application servers are secure. Only traffic that is authenticated, authorized and audited can reach the Secure Zone.

Technical Example

Approaches to integrating maXecurity with your existing infrastructure

The goal of the functional architecture described in the preceding section is for a maXecurity appliance to look like the web server to a web browser, and to look like a web browser to the web server. By working seamlessly with both, the maXecurity appliance transparently enforces your company's security policy. For this architecture to remain secure, it is imperative that the maXecurity appliances intercept, authenticate, and authorize *all* web requests (HTTP/S) before passing them on to the back-end servers, creating the Secure Zone depicted in the previous diagram. It is also necessary to secure the policy store, which holds credentials and other critical configurations.

There are a number of approaches you can employ to achieve this secure architecture within your company's existing infrastructure. Four methods are detailed in this deployment guide. Specifically:

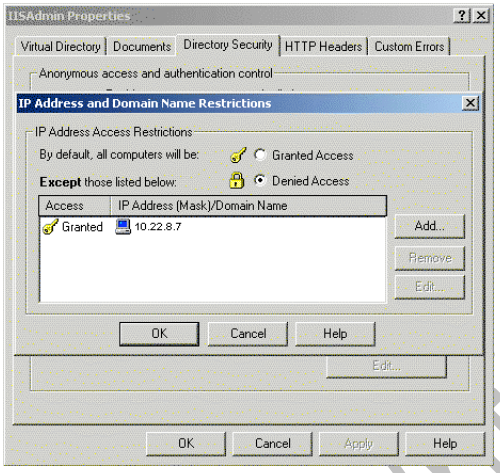
1. **Web Server Access Control Lists (ACLs)** – This approach works by configuring your protected web servers to accept HTTP/S requests only from the IP address of the maXecurity appliances, and/or exclusively accepting requests using the digital certificates installed on the appliances.
2. **Application-layer Firewall** – This approach involves setting up a special network segment (Web services network). Though the back-end servers will remain connected to your company's internal network for local traffic, they will receive web requests only from the maXecurity appliance.
3. **Network Firewalling** – Another method involves employing a network-level firewall (Cisco PIX, Check Point, SonicWALL, etc.) so that the only way to get to the Web services network is from the DMZ network. The maXecurity appliances must be the only devices on the DMZ network for this to function properly.
4. **Hybrid Approach** – This last approach is a combination of the application-layer and network firewalls. The network firewall permits only certain clients to get to the maXecurity appliance on the DMZ, while the maXecurity appliance acts as an application-layer firewall to allow access to the web servers.

You will also use methods similar to these for securing the policy store.

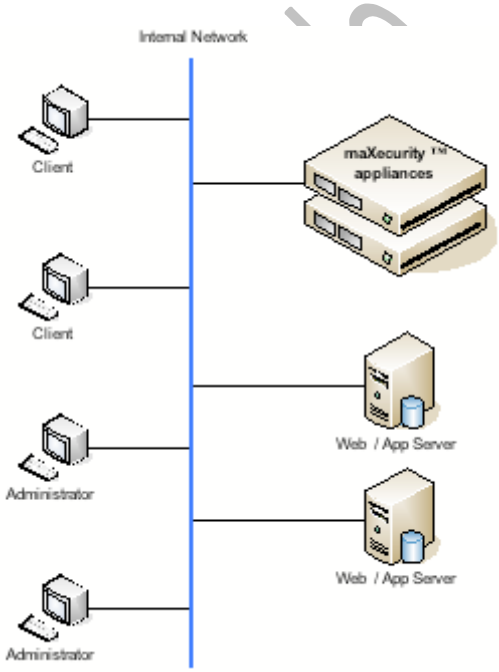
Technical Writing Example

Approach 1: Web Server Access Control Lists

Using Access Control Lists (ACLs) is a straightforward way to ensure that web requests to your back-end servers come exclusively from the maXecurity appliance. It works by configuring your protected web servers to accept requests only from the IP addresses and/or the devices using the digital certificates of the maXecurity appliances.



ACL with Microsoft IIS. Configure the web server to accept connections exclusively from maXecurity appliances to enforce security policies.



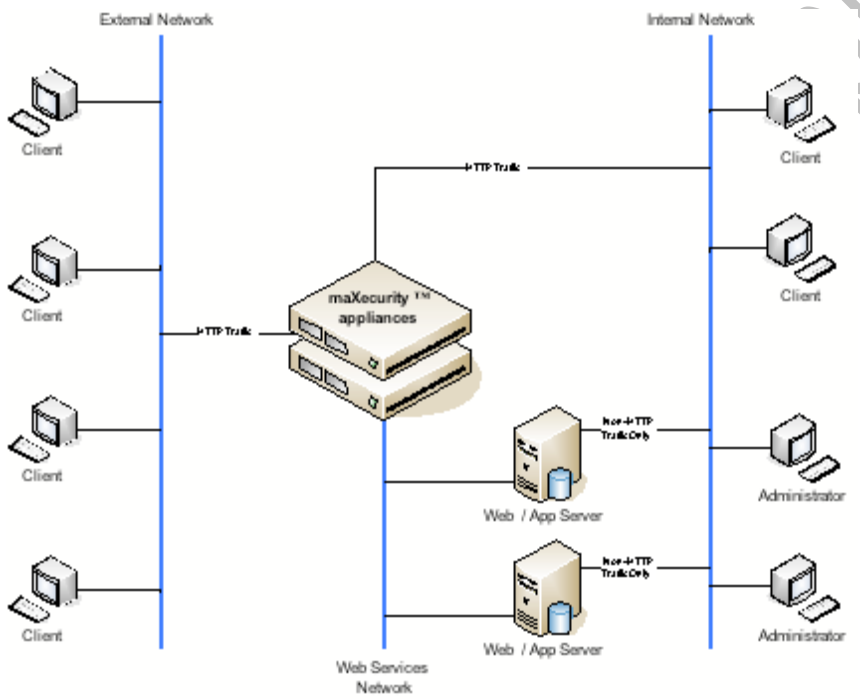
Depending on your web server software (Microsoft IIS, Apache, Sun, etc.) the steps involved may differ slightly, but the approach is identical. Client workstations, administrator workstations, maXecurity appliances and web/applications servers all reside on the same internal network. However, each web server's access control list is restricted to allow only maXecurity appliances to make HTTP/S requests to the web/application servers.

ACLs may be configured using IP address, or by SSL mutual authentication. All maXecurity appliances have a unique X.509 certificate.

Technical Writer

Approach 2: Using maXecurity as an Application-layer Firewall

Each maXecurity appliance has four Gigabit Ethernet interfaces. Each can be set up to connect to different networks. This approach involves setting up a dedicated "Web services network," which the maXecurity appliances are connected to. All web/application servers are then configured to listen only for HTTP/S requests on the special Web services network. The result is that the only way for clients on the external network to access web resources is via the maXecurity appliances.

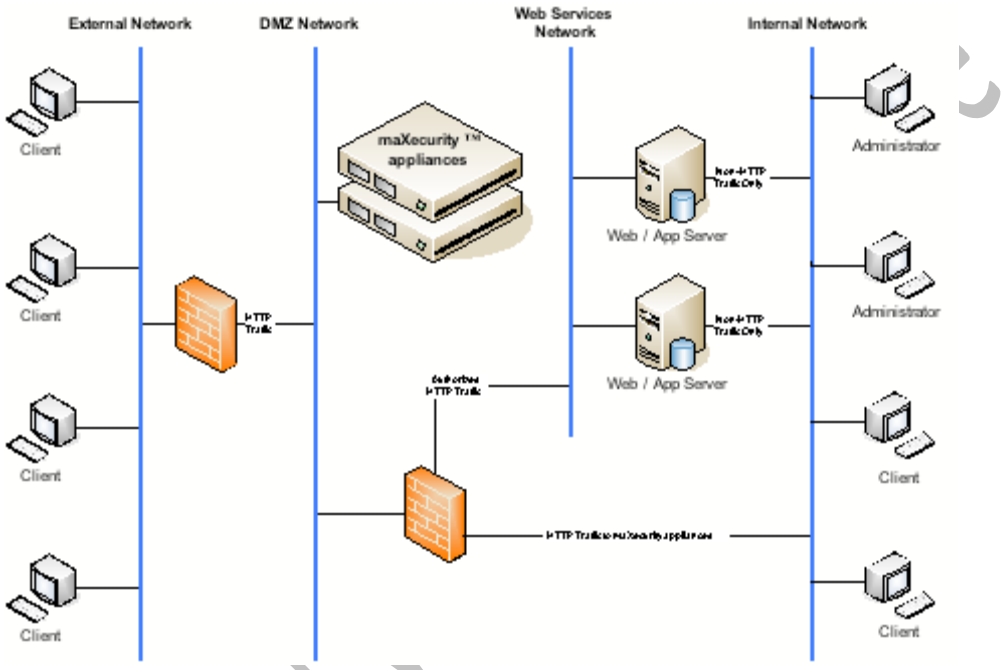


Application-layer Firewall. The only way for external clients, as well as clients and administrators on the internal network, to access web resources is via the maXecurity appliances.

For administration purposes, maXecurity appliances and web/application servers are also connected to the internal network. As you can see from the diagram, though the back-end servers will remain connected to your company's internal network for local traffic, they will receive web requests only from the maXecurity appliance.

Approach 3: Network Firewalling

Another method involves employing a network-level firewall (Cisco PIX, Check Point, SonicWALL, etc.) so that the only way to reach the Web services network is from the DMZ network. The maXecurity appliances must be the only devices on the DMZ network for this to function properly.



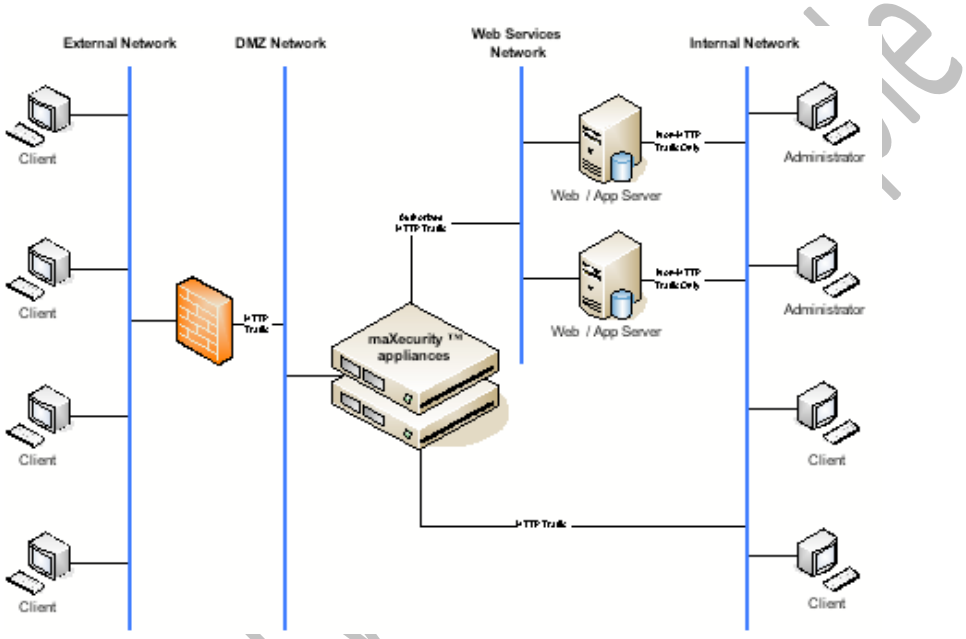
At least four networks must be set up for this approach to function: **External network**, **DMZ network**, dedicated **Web services network**, and the **internal network**.

A network firewall is connected to all four networks. External clients are connected to the external network only. The maXecurity appliances are connected to the DMZ network only. Web/application servers are connected to the Web services network, and are configured to listen for HTTP/S requests on this network only. On the internal network, internal clients and administrators are connected as usual. For administration purposes, maXecurity appliances and web/application servers are also connected to the internal network.

The network firewall is configured to allow connections to the Web services network only from the DMZ network; all other connection attempts for Web services will be denied. The network firewall must also be configured to allow connections to the DMZ network from both the external or internal networks.

Approach 4: Hybrid Application-layer and Network Firewalling

This last approach is a combination of application-layer and network firewalling. The network firewall permits only certain clients to get to the maXecurity appliances on the DMZ, while the maXecurity appliances act as application-layer firewalls to allow access to the web servers. This approach has the advantage of sharing responsibility between the network firewall and the DMZ.



At least four networks must be set up for this approach to function: **External network**, **DMZ network**, dedicated **Web services network**, and the **internal network**.

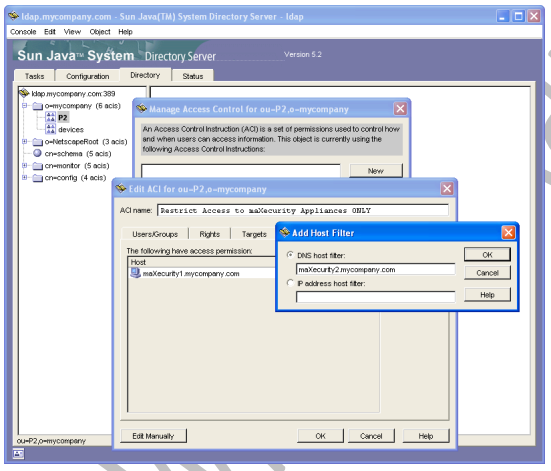
A network firewall is connected to the external network, DMZ network and internal network. Note that in this setup, the network firewall is *not connected* to the Web services network. External clients are connected to the external network. Web/application servers are connected to the Web services network, and are configured to listen for HTTP/S requests on this network only.

Internal clients and administrators remain connected to the internal network. For administration purposes, maXecurity appliances and web/application servers are also connected to the internal network. The network firewall is configured to allow connections to the DMZ network from the external or internal networks. The only way for clients on the external network, as well as clients and administrators on the internal network, to access web resources is via the maXecurity appliances.

Securing the Policy Store

Recall that the policy store is an LDAPv3-compliant directory that stores information about which domains to protect, policies to enforce, and other important configuration information. Because the maXecurity appliance connects immediately to the policy store when it boots, it is imperative that the policy store be secured as well.

The simplest approach is to use Access Control Lists. In the same way we used ACLs to restrict access to web servers, we can do the same on the LDAP directory server.



Securing the policy store with ACLs. The only LDAP client that should be allowed to write to the policy store is the maXecurity appliance.

This can be accomplished using the appliance's IP address, using its BIND DN and password, or by requiring SSL mutual authentication. Remember, all maXecurity appliances have a unique X.509 certificate. For additional security, you may choose to use a network-level firewall as well.

Technical White Paper Sample

Advanced Deployment Techniques

Deploying maXecurity within your company infrastructure ensures the security of your protected web resources. However, there are many other advantages to using maXecurity appliances in conjunction with your web applications. This section details some of the special features built into the maXecurity product suite, including **TrustMapping**, **Authentication Sets**, **User Groups** and **Trusted Cookies**. Keeping these features in mind will allow you to structure your domains more efficiently and incorporate security enhancements into new or existing web applications that your company relies on.

Enhancing Security with Multiple Authentication Methods

One way to enhance the security of your web applications is by using multiple authentication methods. Ordinarily this would result in extra logins that make using the application difficult for your day-to-day users. Using the TrustMapping and digital certificates features built into maXecurity appliances makes this process easier and more efficient.

Feature Focus: TrustMapping™ and Authentication Sets

Authentication Sets: With maXecurity, you have the flexibility to set up as many web sites as you need, each with a different look, feel and security process. An authentication set is a collection of web pages and user login configurations that can be reused across your company infrastructure.

TrustMapping™ is an innovation built into maXecurity products that allows you to set up a trust relationship between two user stores. Security policies from one user store are *trusted* by the other and therefore will not force a user to log in a second time.

By way of example, let's assume you are a brokerage house with an online stockbroker application utilized by all your employees. All users within the company must log in initially in order to view customer accounts and stock performance. This is the most basic level of security. When it comes time to place a trade, however, you want an additional layer of security to ensure that only authorized stockbrokers will be able to make a trade. You must be certain that these users really are who they say they are, and prevent unauthorized use from a stolen account.

Using maXecurity appliances allows you to create a policy on the root directory to authorize users to the **Employee User Store**. This should allow all authenticated users to access all areas of the application. Next, you'll create a subdirectory policy on **/trading**

Technical Writing Example

that uses a different Authentication Set, perhaps this time one that takes advantage of certificate authorization, from the **StockBroker User Store**. Normally when using two user stores you would want to utilize maXecurity's TrustMapping feature to prevent the user from logging in twice. In this case, however, because you are dealing with a sensitive portion of the web application, you'll want to be sure that the StockBroker User Store does *not* implicitly trust the Employee User Store. Instead, maXecurity will require the additional credentials (or a digital certificate) for trading. You can, of course, set the Employee User Store to trust the StockBroker User Store, meaning that a broker who has logged in to place a trade will not have to log in a second time just to look up account details.

Flexible Authorizations with Multiple Security Zones

Another advantage to the maXecurity firmware is the ability to administer flexible authorizations with varying levels of security for different subdirectories (or even individual pages) on your domains.

Feature Focus: User Groups

User Groups: Like authentication sets, maXecurity products let you reuse common user group configurations in User Groups. You can define and assign users by employee type, office location, or any other attribute available in the user stores.

Consider an online record store example application at **www.myrecordstore.com**. The **/admin** subdirectory is protected to allow only employees to access protected resources. By logging in, these admins can edit album descriptions, genres or change pricing information. Let's assume now that we've hired a consultant who will go through and modify our album descriptions to appeal to a younger audience. We'll grant the consultant entry to the **/admin** subdirectory to make these changes, but we do not want to give an outside contractor access to the credit card information also stored in the **/admin** subdirectory, specifically within the page **viewCreditCardDetails.jsp**.

With maXecurity, we can create an additional subdirectory policy for just the single web page **viewCreditCardDetails.jsp**. This policy, for example, could have a different **User Group** that authenticates users whose *employeeType = EMPLOYEE* (as opposed to, say, *CONSULTANT)*. With our new security policy in place, the consultant can access any page under **/admin** but not **/admin/viewCreditCardDetails.jsp**, which requires a user to be in the *EMPLOYEE* User Group.

Technical Writing Example

Integration with 3rd-party Web Applications

Once you have created the Secure Zone with your maXecurity architecture, you will be able to extend your single sign-on, authentication, authorization, and auditing capabilities to integrate with other third-party web applications—even those that do not offer any integration points. The method you use will depend on whether the application supports Application Programmable Interfaces (APIs).

Integrating with 3rd-party Web Application APIs

Feature Focus: Trusted Cookies

User Groups: Trusted cookies allow you to apply fine-grained authorization to your web sites, such as only showing data from a database that correlates with an attribute such as "office location."

Some third-party web applications, such as SAP or WebSphere Portal, have the ability to extend their authentication and authorization beyond the products' built-in support by exposing APIs (Application Programmable Interfaces). The maXecurity appliances can pass secure information to these APIs using a feature called Trusted Cookies. Trusted cookies are cookies sent securely to the web server (not the browser), making it both simple and safe for the web programmer/developer to retrieve useful information from your company's LDAP directories and pass it along to the web application.

By way of example, suppose you are developing a portal using a third-party application such as WebSphere Portal, which allows the use of TAIs (Trust Association Interceptor). In order to integrate maXecurity with one of these web applications, you would create a policy to protect the portal, and add a Trusted Cookie to inject information, such as the *uid* attribute, with a cookie name of *"username"*. The TAI could then be written to read the Trusted Cookies when the user authenticates, and pass on the *uid* attribute to WebSphere Portal, requiring a minimal amount of code:

```
import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.util.Properties;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.ibm.websphere.security.WebTrustAssociationException;
import com.ibm.websphere.security.WebTrustAssociationFailedException;
```

```
import com.ibm.wsspi.security.tai.TAIResult;
import com.ibm.wsspi.security.tai.TrustAssociationInterceptor;

public class maXecurityTAI implements TrustAssociationInterceptor
{

	private String username = null;

	/*
	 * (non-Javadoc)
	 *
	 * @see
com.ibm.wsspi.security.tai.TrustAssociationInterceptor#isTargetInterceptor(javax.servlet.http.HttpServletRequest)
	 */
	public boolean isTargetInterceptor(HttpServletRequest req) throws WebTrustAssociationException
	{
		username = getUsernameCookie(req.getCookies());

		return (username != null);
	}

	private String getUsernameCookie(Cookie[] cookies)
	{
		String cookieName = "username"; // The name of the P2 Trusted Cookie
		if (cookies == null)
			return null;

		for (Cookie cookie : cookies)
			if (cookie.getName().equals(cookieName))
				try
				{
					return URLDecoder.decode(cookie.getValue(), "UTF-8");
				}
				catch (UnsupportedEncodingException e)
				{
					return null;
				}
		return null;
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see
com.ibm.wsspi.security.tai.TrustAssociationInterceptor#negotiateValidateandEstablishTrust(javax.servlet.http.HttpServlet
Request,
	 *      javax.servlet.http.HttpServletResponse)
	 */
	public TAIResult negotiateValidateandEstablishTrust(HttpServletRequest req, HttpServletResponse resp)
throws WebTrustAssociationFailedException
	{
		return TAIResult.create(HttpServletResponse.SC_OK, username);
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#initialize(java.util.Properties)
	 */
	public int initialize(Properties arg0) throws WebTrustAssociationFailedException
	{
		return 0;
```

Technical Writing Example

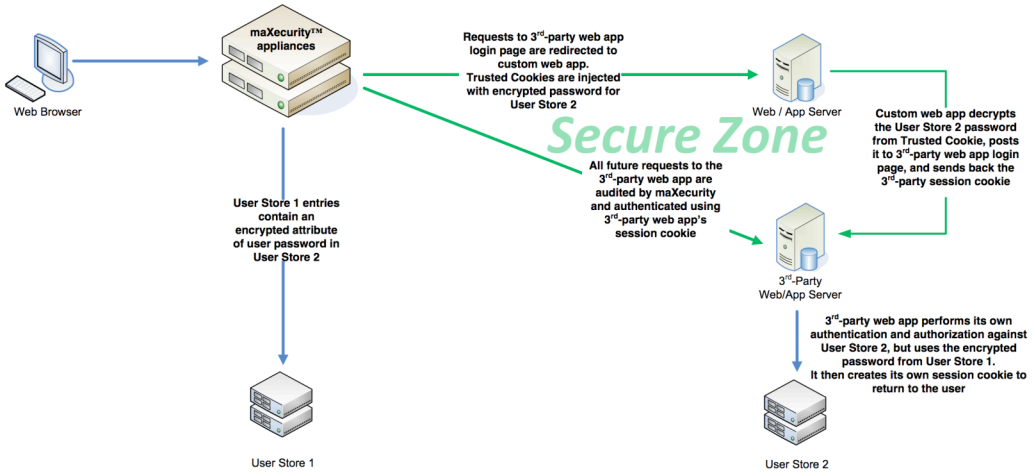
```
}

/*
 * (non-Javadoc)
 *
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#getVersion()
 */
public String getVersion()
{
        return "1.0";
}

/*
 * (non-Javadoc)
 *
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#getType()
 */
public String getType()
{
        return "maXecurity TAI 1.0";
}

/*
 * (non-Javadoc)
 *
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#cleanup()
 */
public void cleanup()
{
        ;
}
```

Integrating with 3rd-party Web Applications without APIs

Some third-party web applications do not allow any extensibility when it comes to security, forcing users to maintain an additional set of credentials to log in. However, even under these circumstances, there are still ways to force secure authentication through the maXecurity product.

The first step is to protect the web application with maXecurity, including the application's login page (see the section on Network Architectures for more information on how to protect back-end web servers and bring them into the Secure Zone). Next, in order to pass the responsibility of securing and auditing access to maXecurity, you must set all user credentials in the third-party application to a random password, and additionally encrypt those random passwords to store them in a maXecurity-accessible User Store. Finally, you will need to create a small web application that can read the encrypted, random password from a Trusted Cookie, decrypt it, and authenticate to the third-party application on behalf of the user.

Technical Writing Example



Integrating with 3rd-party Web Applications without APIs. This process extends the Secure Zone created by maXecurity, as all web aspects of the third-party application are now accessible only to users who successfully authenticate to maXecurity.

From the above diagram, you can see that the first time a web request is made to the application, the third-party login information from maXecurity is passed to the small web app, which passes it along to the third-party web application. This process is completely based upon a trusted connection from maXecurity. When the third-party application sends back *its* session cookie, maXecurity will additionally send its own session cookie to the browser, which keeps the user logged in to both applications for the duration of their sessions. All future web requests to the third-party web application will be audited by maXecurity and passed on to the third-party application along with its own session cookie.

Learning more

Visit *http://www.maXecurity.com* and click on **Support**. If your company has a support contract with P2 Security, you can receive access to a dedicated support representative who is familiar with your web infrastructure and the use of maXecurity products. Support is available 24/7.

Technical Writing Exa